A Distributed Newton Method for Dynamic Network Utility Maximization with Delivery Contracts*

Ermin Wei†, Asuman Ozdaglar†, Atilla Eryilmaz‡, Ali Jadbabaie§

***Abstract*— The standard Network Utility Maximization (NUM) problem has a static formulation, which fails to capture the temporal dynamics in modern networks. This work considers a dynamic version of the NUM problem by introducing additional constraints, referred to as delivery contracts. Each delivery contract specifies the amount of information that needs to be delivered over a certain time interval for a particular source and is motivated by applications such as video streaming or webpage loading. The existing distributed algorithms for the Network Utility Maximization problems are either only applicable for the static version of the problem or rely on dual decomposition and first-order (gradient or subgradient) methods, which are slow in convergence. In this work, we develop a distributed Newton-type algorithm for the dynamic problem, which is implemented in the primal space and involves computing the dual variables at each primal step. We propose a novel distributed iterative approach for calculating the dual variables with finite termination based on matrix splitting techniques. It can be shown that if the error level in the Newton direction (resulting from finite termination of dual iterations) is below a certain threshold, then the algorithm achieves local quadratic convergence rate to an error neighborhood of the optimal solution in the primal space. Simulation results demonstrate significant convergence rate improvement of our algorithm, relative to the existing first-order methods based on dual decomposition.**

I. Introduction

The emergence and widespread use of large scale communication networks motivated much recent research interest in developing distributed methods for solving resource allocation problems over networks. One of the important tasks in the wireline networks is to allocate the available bandwidth among the sources, which can be formulated as a *Network Utility Maximization* problem (referred to as the *NUM* problem in the literature ([3], [7], [1], [8], [12]). Since first proposed in [6], this formulation has been used extensively in the analysis of current wireline network congestion control protocols and design of new wireline and wireless network protocols. A standard Network Utility Maximization problem consists of a fixed set of sources with predetermined routes. Each source has a local utility function, which is a function of its source rates, and the system utility is defined as the sum of the individual utilities. The sources choose their source rates to collectively maximize the system utility, while respecting the link capacity constraints. Existing works focus on solving this problem by distributed methods that use either dual decomposition and first-order (subgradient) algorithms ([6],[8], [10]) or second order Newton-type algorithm ([14], [15]). First order methods employ a simple dual price exchange mechanism but suffer from slow rate of convergence. Second order algorithms have much faster rate of convergence, but involve more complex computations that require global information. The recent paper ([14] proposed a Newton type method in which computations can be implemented using local information.

The standard formulation treats the network as static and fails to capture the temporal dynamics in modern networks. In video streaming and page loading applications, for instance, certain amount of information has to be delivered within some small time intervals in order to meet quality of service constraints, which can be viewed as *delivery contracts*. In this paper, we focus on a dynamic version of the NUM problem. We adopt the multi-period NUM formulation introduced in [13], which used delivery contracts to couple the source rates across time. Based on dual decomposition techniques, the authors in [13] develop a distributed algorithm for this problem when all the problem parameters are known a priori. When the parameters of the problem are not known, the authors suggest a heuristic based on model predictive control to approximately solve the problem. The method is, however, slow in convergence.

In this paper, we propose a Newton-type distributed second order method for dynamic NUM problem with delivery contracts, which can be used both when the parameters are known before hand and in the model predictive control heuristic. The proposed algorithm is an extension of the distributed Newton algorithm for NUM problem developed in [14], where at each Newton step the dual variables are computed in a decentralized manner using some novel finitely terminated iterative scheme based on matrix splitting techniques. By using a similar analysis as in [15], we can show that if the error due to finite truncation of the dual iterations satisfy certain error tolerance level, then the overall algorithm converges quadratically to an error neighborhood of the optimal function value. The size of the error neighborhood depends explicitly on the error tolerance level.

*This research is supported by NSF Career grant DMI-0545910, AFOSR MURI R6756-G2, ONR MURI N000140810747, and AFOSR Complex Networks Program.

†Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

‡ Electrical and Computer Engineering Ohio State University

§Department of Electrical and Systems Engineering and GRASP Laboratory, University of Pennsylvania.

The rest paper is organized as follows: Section II defines the formulation of the problem and introduces some related transformations. Section III briefly describes the standard centralized Newton method, where each update depends on the value of the dual variables. Section IV specifies one distributed approach to compute the dual variables based on matrix splitting technique. Section V presents a procedure to calculate the inexact Newton primal update using the computed dual variables. Section VI shows some simulation results to demonstrate convergence speed improvement of our algorithm over the existing distributed method. Section VII contains our concluding remarks.[1]

Basic Notation and Notions:

For a matrix A, we use $[A]_i$ to denote the i^{th} column of the matrix A, and $[A]^j$ to denote the j^{th} row of the matrix A. We denote by A' the transpose of matrix A. We write $I(n)$ to denote the identity matrix of dimension $n \times n$ and $0(n \times m)$ to denote the zero matrix of size $n \times m$. A real-valued convex function $g : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}$, is *self-concordant* if $|g'''(x)| \leq 2g''(x)^{\frac{3}{2}}$ for all x in its domain.[2] For real-valued functions in $\mathbb{R}^n$, a convex function $g : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}^n$, is self-concordant if it is self-concordant along every direction in its domain. (see [2] Chapter 9 for more details).

II. Dynamic Network Utility Maximization Problem with Delivery Contracts

We consider a dynamic version of the network utility maximization problem with finite time horizon, time-varying network topology and delivery contracts, which was first studied in [13]. The network underlying this problem consists of a set $\mathcal{L} = \{1, ..., L\}$ of (directed) links and a set $\mathcal{S} = \{1, ..., S\}$ of sources. We let t denote the time index, which takes value from the set $\{1, 2, \ldots, T\}$. We use positive scalar $\bar{c}_l^t$ to denote the finite link capacity for link l at period t. At each time period t, the link capacities can be expressed by the vector $\bar{c}^t = [\bar{c}_l^t]_{l \in \mathcal{L}}$. At each time period, each source transmits information flow along a predetermined route.[3] For each link l, the set $S^t(l)$ consists of the set of sources using it at period t. Similarly, for each source i, the set $L^t(i)$ is the set of links it uses at period t. We use nonnegative scalar s_i^t to denote the source rate for source i at period t, the vector $s^t = [s_i^t]_{i \in \mathcal{S}}$ to refer to the nonnegative source rate vector. We define the $(L \times S)-$matrix $\bar{R}^t$ to be the *routing matrix*, given by

$$\bar{R}_{ij}^t = \begin{cases} 1 & \text{if link } i \text{ is on the route of source } j \text{ at period } t, \\ 0 & \text{otherwise.} \end{cases}$$

The aggregate flow on a link l during period t can be represented as $[\bar{R}^t s^t]_l$. Hence the set of link capacity constraints can be written as $\bar{R}^t s^t \leq \bar{c}^t$ for all t.

A delivery contract can be imposed on a source i, specifying the required minimal flow $\bar{q}$ to be delivered during a time interval $[\underline{t}, \bar{t}]$,[4] i.e., $\sum_{t=\underline{t}}^{\bar{t}} s_i^t \geq \bar{q}$. We say a contract is *active* at periods t if t is in the corresponding contract interval $[\underline{t}, \bar{t}]$. Each source i is subject to m_i such delivery contracts, with $\bar{q}_i$ in $\mathbb{R}^{m_i}$ denoting the associated delivery contract amount. We use the $(m_i \times T)-$matrix $\bar{E}_i$ to represent the *delivery contract indicator matrix*, given by

$$[\bar{E}_i]_{jt} = \begin{cases} 1 & \text{if the } j^{th} \text{ contract is active at period } t, \\ 0 & \text{otherwise.} \end{cases}$$

With the above notation, for each source i the delivery contract requirement can be written as $\bar{E}_i s_i \geq \bar{q}_i$, where s_i denotes the source rate allocation for source i, i.e., $s_i = [s_i^t]_{t \in \{1,2,\ldots,T\}}$.

For each source i at each time period t, we associate a utility function $U_i^t : \mathbb{R}_+ \to \mathbb{R}$. The overall utility of the problem is the sum of all utility functions over the entire time horizon. The *dynamic Network Utility Maximization (NUM) with delivery contracts* problem, can be formulated as

$$\begin{aligned} \text{maximize} \quad & \sum_{t=1}^{T} \sum_{i=1}^{S} U_i^t(s_i^t) && (1) \\ \text{subject to} \quad & \bar{R}^t s^t \leq \bar{c}^t, \quad s^t \geq 0, && \text{for all } t \text{ in } \{1, 2, \ldots, T\}, \\ & \bar{E}_i s_i \geq \bar{q}_i, && \text{for all } i \text{ in } \mathcal{S}, \end{aligned}$$

where the utility functions satisfy the following assumptions.

Assumption 1: The utility functions $U_i^t : \mathbb{R}_+ \to \mathbb{R}$ are additive, continuous, strictly concave, monotonically nondecreasing on $\mathbb{R}_+$ and twice continuously differentiable on the set of positive real numbers. The functions $-\sum_{i=1}^{S} U_i^t : \mathbb{R}_+ \to \mathbb{R}$ are self-concordant on the set of positive real numbers for all periods $t = 1, \ldots, T$.

The self-concordance property is satisfied by many utility functions considered in the literature, α-fair utility functions with $\alpha = 1$ for instance [9] and is adopted here to establish local quadratic convergence rate. The above formulation can be used when all the problem parameters are known a priori. In [13], the authors also introduce a model predictive control based heuristic to approximately solve an online version of the problem where the link capacities $\bar{c}^\tau$ are only known at time $t \geq \tau$. In the modified problem, both the time horizon and delivery contract constraints are adjusted to reflect the remaining time period and the predicted capacity given the realized values are used instead of $\bar{c}^\tau$. In the modified formu-

[1] Due to space constraints we omit some of the proofs here, interested readers can find the full version of the paper at http://www.mit.edu/∼erminwei/Publication.html.

[2] Self-concordant functions are defined through the following more general definition: a real-valued convex function $g : X \to \mathbb{R}$, where X is a subset of $\mathbb{R}$, is *self-concordant*, if there exists a constant $a > 0$, such that $|g'''(x)| \leq 2a^{-\frac{1}{2}} g''(x)^{\frac{3}{2}}$ for all x in its domain [11], [5]. Here we focus on the case $a = 1$ for notational simplification in the analysis.

[3] The routes can be time-varying. To avoid the trivial cases, we assume that at each time period, each source flow traverses at least one link and each link is used by at least one source and at each time the (undirected) links form a connected graph.

[4] Our algorithm works in the case when delivery contract is defined over non-consecutive time periods also. This setting was chosen for convenience.

lation, the problem structure remains the same, and therefore, in this paper, we focus on developing a fast distributed algorithm for Problem (1).

For notational simplicity, we let vector $s = [s_1', s_2', \ldots, s_S']'$ in $\mathbb{R}^{TS}$ be the source rate variables, matrix R in $\{0,1\}^{TL \times TS}$ be the corresponding routing matrix for all time periods given by $R = \begin{bmatrix} [\bar{R}^1]_1 & 0\,(L \times (T-1)) & \cdots & 0\,(L \times (T-1)) \\ 0\,(L \times 1) & [\bar{R}^2]_1 & \cdots & 0\,(L \times (T-2)) \\ \vdots & \vdots & \ddots & \vdots \\ 0\,(L \times T-1) & [\bar{R}^T]_1 & \cdots & [\bar{R}^T]_S \end{bmatrix}$. We define the block diagonal matrix E in $\{0,1\}^{M \times TS}$ to be the aggregate delivery contract matrix for all sources, i.e., $E = \begin{bmatrix} \bar{E}_1 & 0(m_2 \times T) & \cdots & 0(m_S \times T) \\ 0(m_1 \times T) & \bar{E}_2 & \cdots & 0(m_S \times T) \\ \vdots & \vdots & \ddots & \vdots \\ 0(m_1 \times T) & 0(m_2 \times T) & \cdots & \bar{E}_S \end{bmatrix}$, where M is the total number of delivery contracts, i.e., $M = \sum_{i=1}^{S} m_i$. Set $E^t(i)$ denotes the set of active constraints for source i at time period t, i.e., $E^t(i) = \{m \mid E_{m((i-1)T+t)} = 1, 1 \le m \le M\}$. Set $\psi(m)$ denotes the set of active periods for contract m. We use the aggregated capacity vector c of length (TL) to denote $c = [(\bar{c}^1)', (\bar{c}^2)', \ldots, (\bar{c}^T)']'$ and contract amount vector q of length M to denote $q = [q_1', q_2', \ldots, q_S']'$. By using the new notation, Problem (1) can written compactly as

$$\begin{aligned} \text{maximize} \quad & \sum_{t=1}^{T} \sum_{i=1}^{S} U_i^t(s_i^t) \\ \text{subject to} \quad & Rs \le c, \quad s \ge 0, \\ & Es \ge q. \end{aligned} \tag{2}$$

To facilitate the development of a distributed Newton method, we employ a similar interior-point approach as in [14] and reformulate the problem into one with only equality constraints by using slack variables. For the link capacity constraints, we introduce a length (TL) vector y of nonnegative slack variables, i.e., $Rs + y = c$, and denote by $[y_l^t]$ the slack in capacity constraint of link l at period t. Similarly, we associate a vector of nonnegative slack variables z in $\mathbb{R}^M$ for the delivery contracts, with z^m representing the slack variable associated with the m^{th} delivery contract. Hence the delivery contract constraints can be rewritten as $Es - z = q$.

We then impose logarithmic barrier functions for the nonnegativity constraints. The new decision vector is $x = [s', y', z']'$ and we use notation x_i^t to refer to the rate of source i during period t, i.e., $x_i^t = x_{S(i-1)+t} = s_i^t$. Problem (2) can be rewritten as

$$\begin{aligned} \text{minimize} \quad & -\sum_{t=1}^{T} \sum_{i=1}^{S} U_i^t(x_i^t) - \mu \sum_{j=1}^{TS+TL+M} \log(x_i) \\ \text{subject to} \quad & Ax = b \end{aligned} \tag{3}$$

where μ is a positive coefficient for the barrier functions, vector $b = \begin{bmatrix} c' & q' \end{bmatrix}'$ and A is a matrix of dimension $(TL + M) \times (TS + TL + M)$. Matrix A is defined by $A = \begin{bmatrix} F' & G' \end{bmatrix}'$, where $F = [R \quad I(TL) \quad 0(TL \times M)]$ and $G = [E \quad 0(M \times TL) \quad I(M)]$. We denote by $f : \mathbb{R}_+^{TS+TL+M} \to \mathbb{R}$ the objective function, i.e.,

$$f(x) = -\sum_{t=1}^{T} \sum_{i=1}^{S} U_i^t(x_i^t) - \mu \sum_{j=1}^{TS+TL+M} \log(x_i),$$

and by f^* the optimal value for the equality constrained problem (3). Due to the result from [15], which shows a problem of the form Problem (2) can be addressed by solving two instances of Problem (3) with different coefficients $\mu \ge 1$, in the rest of the paper we focus on Problem (3) with $\mu \ge 1$.

III. Exact Newton Method

Our distributed Newton method for problem (3) is developed based on a (feasible start) equality-constrained Newton method (see [2] Chapter 10). In our iterative method, we use $x(k)$ to denote the primal vector at the k^{th} iteration.

To initialize the algorithm, we start with some feasible and strictly positive vector $x(0) > 0$. We assume such feasible initialization is easy to find. This is the case in many applications, where the capacity constraints are relatively large compared to the delivery contract constraints.

We denote the Hessian matrix by $H_k = \nabla^2 f(x(k))$ for notational convenience. Given an initial feasible vector $x(0)$, the algorithm generates the iterate sequence by

$$x(k+1) = x(k) + s(k)\Delta x(k),$$

where $s(k)$ is a positive stepsize. The vector $\Delta x(k)$ is the (primal) Newton direction given as the solution to the following linear system of equations

$$\Delta x(k) = -H_k^{-1}\left(\nabla f(x(k)) + A'w(k)\right), \text{ and} \tag{4}$$

$$(AH_k^{-1}A')w(k) = -AH_k^{-1}\nabla f(x(k)), \tag{5}$$

where $w(k)$ in $\mathbb{R}^{(TL+M)}$ is the dual vector associated with the equality constraints. The first TL elements of $w(k)$ are associated with the link capacity constraints for each time period and the last M elements are for each of the delivery contracts. Direct computation of $w(k)$ involves the evaluation of the matrix inverse $(AH_k^{-1}A')^{-1}$, which is both costly and requires global information, therefore cannot be implemented in a decentralized way. In the next section, we present an iterative approaches to compute the dual vector $w(k)$ in a distributed manner based on matrix splitting technique.

IV. Distributed Dual Variable Computation

In Section IV-A, we introduce some notations, a key lemma and specify what operations are allowed in the distributed algorithm. In Section IV-B, we develop the iterative decentralized algorithm based on matrix splitting technique. We use $w(k,n)$ to denote the dual variable value at the n^{th} dual iteration at the k^{th} primal step. We use $w_l^t(k,n) =$

$w_{l+(t-1)L}(k,n)$ to denote the dual variable of $w(k,n)$ corresponding to the l^{th} link at time period t and notation $w_{TL+m}(k,n)$ to refer to the dual variable associated with the m^{th} delivery contract.

A. Preliminaries

For notational convenience, we introduce the following functions:

(A) Functions $t(\cdot) : \{1,\dots,TL\} \to \{1,2,\dots,T\}$ and $l(\cdot) : \{1,\dots,TL\} \to \{1,2,\dots,L\}$ are defined as, for v in $\{1,\dots,TL\}$, $t(v) = \lceil \frac{v}{L} \rceil$, the time period of v and $l(v) = v \mod L$, the link associated with v.

(B) Functions $s(\cdot) : \{1,\dots,TS\} \to \{1,2,\dots,S\}$ and $\tau(\cdot) : \{1,\dots,TS\} \to \{1,2,\dots,T\}$ are defined as, for u in $\{1,\dots,TS\}$ $s(u) = u \mod T$, the time period of u, $\tau(u) = \lceil \frac{u}{T} \rceil$, the source associated with u.

(C) Function $i(\cdot) : \{1,2,\dots M\} \to \{1,2,\dots S\}$ is given by $i(m) = \max_{i\in\mathbb{Z}} \left\{ \sum_{j=0}^{i-1} m_j \le m \right\}$, i.e., $i(m)$ is the source with which the m^{th} delivery contract is associated.

(D) Function $H_k^{-1}(x_i)$ is defined as $H_k^{-1}(x_i) = [H_k^{-1}]_{ii}$.[5]

(E) Function $\nabla f(x)(x_i)$ is a function of x_i with $\nabla f(x)(x_i) = [\nabla f(x)(x_i)]_i$.

We define *weighted price of the route* for source i at time t, $\pi_i^t(k,n)$, as $\pi_i^t(k,n) = (H_k^{-1})(s_i^t) \sum_{l\in L^t(i)} w_l^t(k,n)$, which is the sum of dual variables associated with the link capacities along the route of source i weighted by the Hessian element associated with that source at time t. Similarly, we denote the *weighted price of the contracts* for source i at time t by $\xi_i^t(k,n)$, which is given by $\xi_i^t(k,n) = (H_k^{-1})(s_i^t) \sum_{j\in E^t(i)} w_{TL+j}(k,n)$, which is the sum of dual variables corresponding to the active delivery contracts of source i at time t weighted by the Hessian element associated with the source. We next define the notion of a distributed algorithm.

Each of the link and source is viewed as a processor. An algorithm is called *distributed* if it only uses the following private knowledge and information exchange scheme, for each primal iteration k and dual iteration n:

(A) Private knowledge:

(A.a) Each source i knows its source rate, routes, utility function and associated barrier function (and first and second derivative thereof) for the entire time horizon, i.e., $s_i^t(k)$, columns $(i-1)S+1$ to iS of the matrix R and $U_i^t(s_i^t(k)) + \mu \log(s_i^t(k))$ for $t = \{1,2,\dots T\}$.

(A.b) Each source i knows its own delivery contract(s), their corresponding dual variable(s) and slack variable(s), i.e., $\bar{E}_i$, q_i, $w_{TL+m}(k,n)$, $z^m(k)$ for all m with $i(m) = i$.

(A.c) Each link l knows its own capacity constraints, slack variable and dual variable associated with the constraint for the entire time horizon, i.e., $[c_l^t]_t$, $y_l^t(k)$, $w_l^t(k,n)$ for $t = \{1,2,\dots T\}$.

(B) Information exchange:

(B.a) Source i may communicate with the links in the set $\cup_t^T L^t(i)$, information at the links is aggregated a long a route and sent back to each source using a feedback mechanism, as shown in Figures 1 and 2.[6]

(B.b) Links may receive information from sources traversing them, i.e. $\cup_t^T S^t(l)$, and perform simple algebraic operations on the data.

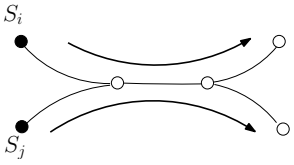

Fig. 1. Direction of information flow from sources to the links they use.

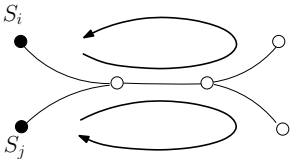

Fig. 2. Direction of flow from links to the sources using them.

The definition of distributed algorithm above implies that the weighted price of the contracts and weighted price of the route for source i at time t, i.e., $\xi_i^t(k,n)$ and $\pi_i^t(k,n)$, can be computed in a distributed way by source i. Also each source can calculate quantity $H_k^{-1}(z^v) w_{TL+v}(k,n)$ for any contract v associated with that source. For link l, the term $H_k^{-1}(y_l^t)\, w_l^t(k,n)$ can be calculated using local information for all period t. The following key lemma expresses the matrix-vector product $AH_k^{-1}A'w(k,n)$ in a form that can be computed in a distributed way. This lemma plays an essential role in enabling us to compute the dual variables in a decentralized way.

Lemma 4.1: The matrix vector product $AH_k^{-1}A'w(k,n)$ can be written in the following form

$$[AH_k^{-1}A'w(k,n)]_v = \sum_{i\in S^{t(v)}(l(v))} [\pi_i^{t(v)}(k,n) + \xi_i^{t(v)}(k,n)] + H_k^{-1}\left(y_{l(v)}^{t(v)}\right) w_{l(v)}^{t(v)}(k,n),$$

for $v = 1,\dots TL$ and

$$[AH_k^{-1}A'w(k,n)]_v = \sum_{t\in\psi(v)} [\pi_{i(v)}^t(k,n) + \xi_{i(v)}^t(k,n)] + H_k^{-1}(z^v) w_{TL+v}(k,n),$$

for $v = 1,\dots,M$.

B. Distributed Computation of Dual Variables I: Matrix Splitting

By using the matrix splitting technique employed in [14] and [15] (see [4] for a comprehensive review), we can

[5]Due to the fact that the objective function is separable, convex, and contains a logarithmic component for each variable, the Hessian matrix H_k^{-1} is strictly positive definite and diagonal. Hence the functions $H_k^{-1}(x_i)$ completely characterize the matrix H_k^{-1}.

[6]We allow communication between sources and all the links it uses in the finite time horizon in this definition. This is required by the temporal coupling nature of the problem and it is reasonable because within a short time horizon the network topology is unlikely to vary much.

develop an iterative way to compute the dual variables. The development is very similar to those in [15] and we highlight the key result in the following theorem.

Theorem 4.2: For each primal iteration k, the dual variables are given as the limit of the following iteration,

$$w_v(k,n+1) = [\bar{D}_k^{-1}]_{vv}\Big([\bar{B}_k]_{vv}w_v(k,n) + [D_k]_{vv}w_v(k,n) - [AH_k^{-1}A'w(k,n)]_v + \sum_{i\in S_{l(v)}^{t(v)}} H_k^{-1}(s_i^{t(v)})\nabla f(x(k))(s_i^{t(v)}) + H_k^{-1}\left(y_{l(v)}^{t(v)}\right)\nabla f(x(k))\left(y_{l(v)}^{t(v)}\right)\Big),$$

for $1 \le v \le TL$ and for $v = m + TL > TL$,

$$w_v(k,n+1) = [\bar{D}_k^{-1}]_{vv}\Big([\bar{B}_k]_{vv}w_v(k,n) - [AH_k^{-1}A'w(k,n)]_v + H_k^{-1}(z^m)\nabla f(x(k))(z^m) + [D_k]_{vv}w_v(k,n)\sum_{t\in\psi(m)} H_k^{-1}(s_{i(m)}^t)\nabla f(x(k))(s_{i(m)}^t)\Big),$$

where matrix D_k is diagonal with diagonal entries $[D_k]_{vv} = [AH_k^{-1}A']_{vv}$, matrix B_k is symmetric, defined by $B_k = AH_k^{-1}A' - D_k$, matrix $\bar{B}_k$ is a diagonal matrix with diagonal entries $[\bar{B}_k]_{vv} = \sum_{j=1}^{TL+M}[B_k]_{ij}$ and the diagonal matrix $\bar{D}_k$ is given by $\bar{D}_k = D_k + \bar{B}_k$.

By using definition of the above matrices and Lemma 4.1, we can verify that all the above dual iteration can be implemented in a distributed way.

V. Distributed Inexact Newton Method

Given a dual vector computed using the method in the previous section, i.e., $w(k) = w(k,n)$ for some finite n, we can now generate the primal update. Relation (4) combined with the diagonal structure of the Hessian matrix H suggests that for $1 \le v \le TS$,

$$[\Delta x(k)]_v = -H_k^{-1}(x_v)\Big(\nabla f(x(k))(x_v) + [A'w(k)]_v\Big), \quad (6)$$

where the last term $[A'w(k)]_v$ can be express as $[A'w(k)]_v = \sum_{l\in L^{\tau(v)}(s(v))} w_l^{\tau(v)}(k)$. Hence the above relation can be computed in a distributed way.

Our distributed Newton method computes the primal Newton direction in two stages. In the first stage, the first TS components of $\Delta\tilde{x}(k)$, are computed using relation (6). These elements correspond to the source rates over the entire time horizon, which we denote by $[\Delta\tilde{s}_i^t(k)]_{i,t}$. In order to maintain feasibility of $Ax = b$ and guarantee all the capacity and delivery contract constraints are satisfied, we calculate separately the last $TL + M$ components of $\Delta\tilde{x}(k)$, corresponding to the slack variables for the capacity and delivery contract constraints, denoted by $[\Delta\tilde{y}_l^t(k)]_{l,t}$ and $[\Delta\tilde{z}^m(k)]_m$. The feasibility correction is given by $[\Delta\tilde{y}_l^t(k)]_{l,t} = (\Delta\tilde{x}(k))_{\{TS+1,\ldots,TS+TL\}} = -R\,(\Delta\tilde{x}(k))_{\{1,\ldots,TS\}}$, and $[\Delta\tilde{z}^m(k)]_m = (\Delta\tilde{x}(k))_{\{TS+TL+1,\ldots,TS+TL+M\}} = E\,(\Delta\tilde{x}(k))_{\{1,\ldots,TS\}}$.

Starting from an initial feasible vector $x(0)$, the *(inexact) distributed Newton algorithm* generates the primal vectors $x(k)$ as follows:

$$x(k+1) = x(k) + s(k)\Delta\tilde{x}(k), \quad (7)$$

where $s(k)$ is a positive stepsize, and $\Delta\tilde{x}(k)$ is the inexact Newton direction at the k^{th} iteration generated as above.

Our stepsize choice is based on the quantity *inexact Newton decrement* $\tilde{\lambda}(x(k))$, which is given by

$$\tilde{\lambda}(x(k)) = \sqrt{(\Delta\tilde{x}(k))'\nabla^2 f(x(k))\Delta\tilde{x}(k)}. \quad (8)$$

Note that $\tilde{\lambda}(x(k))$ is nonnegative and well defined because the matrix $\nabla^2 f(x(k))$ is positive definite. The inexact Newton decrement can be written as a sum of quantities known by individual sources and links. One way to compute this summation exactly is by using a distributed iterative scheme with finite termination described in [14]. Based on the computed value $\tilde{\lambda}(x(k))$, we use a stepsize rule given by

$$s(k) = \begin{cases} \frac{c}{\tilde{\lambda}(x(k))+1} & \text{if } \tilde{\lambda}(x(k)) \ge \frac{1}{4}, \\ 1 & \text{otherwise,} \end{cases} \quad (9)$$

where c is some positive scalar that satisfies $\frac{5}{6} < c < 1$.

In order to achieve quadratic rate of convergence, we impose the following assumption on the errors in this algorithm due to finite truncation of the dual iterations.

Assumption 2: Let $\gamma(k)$ denote the error in the primal Newton direction, i.e., $\Delta x(k) = \Delta\tilde{x}(k) + \gamma(k)$. For all k, $\gamma(k)$ satisfies $|(\gamma(k))'\nabla^2 f(x(k))\gamma(k)| \le p^2(\Delta\tilde{x}(k))'\nabla^2 f(x(k))\Delta\tilde{x}(k) + \epsilon$ for some positive scalars $p < 1$ and ϵ.

By using a similar approach as in [14], it can be shown that the above condition can be achieved and verified in a distributed way. The convergence properties for the distributed Newton algorithm follows directly from the analysis in [15], since matrix A has full row rank and Problem 3 has the exact same formulation as in [15] . Hence it can be shown that the algorithm converges quadratically to an error neighborhood of the optimal solution, where the size of the error neighborhood can be explicitly characterized using the parameters of the algorithm and the error tolerance level at each dual iteration.

VI. Simulation Results

For comparison purposes, we simulated our distributed Newton algorithm for dynamic NUM problem using a setting similar to [13], with time horizon $T = 10$. The network topology is fixed and is shown in Figure 3. We impose the following 4 delivery contracts, $\sum_{t=1}^{3} s_1^t \ge 1$, $\sum_{t=6}^{8} s_1^t \ge 1$, $\sum_{t=3}^{6} s_2^t \ge 1.2$ and $\sum_{t=3}^{10} s_3^t \ge 1.2$. The utility functions for each source i is given by $f_i(s_i) = 200\log(s_i + 0.1)$. The constant 0.1 is necessary to guarantee the logarithmic function is numerically stable for small s_i. The capacity for link l at each time t is chosen to be independently uniformly

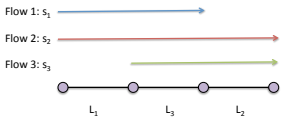

Fig. 3. Network used for simulation, from [13].

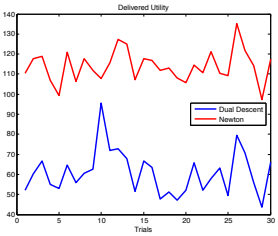

Fig. 4. Utilities generated by two online algorithms over 30 trials with randomly generated capacities each 10-period long. The utilities generated in time frames where there is a delivery contract active and unmet is excluded.

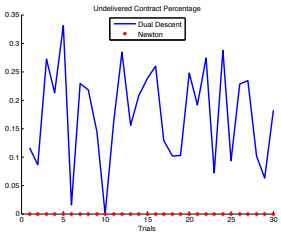

Fig. 5. Percentage of undelivered contracts of two algorithms over 30 trials with randomly generated capacities each 10-period long.

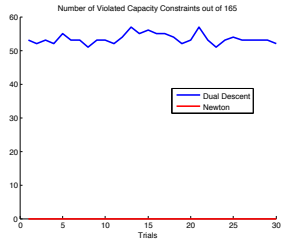

Fig. 6. Number of total capacity constraints violated by two online algorithms over 30 trials with randomly generated capacities each 10-period long.

random between the interval $[[\bar{R}^t]^l s^0, 1]$.[7] We simulated the case where capacity constraints $\bar{c}^\tau$ are known at time $t \geq \tau$ using model predictive control techniques and we use the expected value to forecast $\bar{c}_l^\tau$ for $\tau \geq t$.

To investigate the performance of the distributed Newton algorithm, we simulated 30 trials each 10-period long with randomly generated capacity constraint parameters $\bar{c}_l^t$. For every trial, at each time period, distributed Newton and the dual descent algorithm were both given 0.5 seconds to execute. Figure 4 records the utilities generated by two online algorithms over 30 trials. To enforce the delivery contract constraints, we excluded the utilities associated with source i in time periods when source i had an active delivery contract which was not satisfied by the end of the time horizon. In all trials, distributed Newton method was able to deliver higher utility. In Figure 5, we present the percentage of undelivered contracts over 30 trials, where the percentage is measured as $\sum_{i=1}^{S} \frac{\max\{0, q_i - D_i s_i\}}{q_i}$ at the end of each trial. Since the distributed Newton algorithm is an interior point method, all solutions were feasible and hence all delivery contracts were satisfied, while the dual descent algorithm had about 15% undelivered contract amount. We also analyze the violation of the capacity constraints. In each trial, there were 165 capacity constraints in total. The distributed Newton algorithm remained feasible for all trials, while the dual descent algorithm violates about 55 constraints on average, as shown in Figure 6. We conclude that due to the faster convergence speed and interior point nature of the algorithm, within the same amount of computation time, the distributed Newton method was able to produce a solution that delivers higher utility while satisfying all the constraints.

VII. Conclusions

This paper develops a distributed inexact Newton method for dynamic Network Utility Maximization problems, based on matrix splitting technique to solve for the dual variables. It can be shown that the algorithm converges quadratically to an error neighborhood of the optimal solution, where the size of the error neighborhood can be explicitly characterized using the parameters of the algorithm and the error tolerance level at each dual iteration. Simulations demonstrate significant improvements over the existing distributed algorithm. Our future works include to extend Newton-type fast converging algorithm to not self-concordant utility functions, to develop and investigate performance of other techniques in solving the dual variables, including conjugate gradient type distribued methods, and to develop asynchronous version of the distributed Newton algorithm.

[7] This interval was chosen to make feasible interior initialization simple.

References

[1] S. Athuraliya and S. Low. Optimization flow control with Newton-like algorithm. *Journal of Telecommunication Systems*, 15:345–358, 2000.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] M. Chiang, S. H. Low, A. R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: a mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[4] R. Cottle, J. Pang, and R. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.

[5] F. Jarre. Interior-point methods for convex programming. *Applied Mathematics and Optimization*, 26:287–311, 1992.

[6] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[7] F. P. Kelly, A. K. Maulloo, and D. K. Tan. Rate control for communication networks: shadow prices, proportional fairness, and stability. *Journal of the Operational Research Society*, 49:237–252, 1998.

[8] S. H. Low and D. E. Lapsley. Optimization flow control, I: basic algorithm and convergence. *IEEE/ACM Transaction on Networking*, 7(6):861–874, 1999.

[9] J. Mo and J Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5), 2000.

[10] A. Nedic and A. Ozdaglar. *Convex Optimization in Signal Processing and Communications*, chapter Cooperative distributed multi-agent optimization. Eds., Eldar, Y. and Palomar, D., Cambridge University Press, 2008.

[11] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, 2001.

[12] R. Srikant. *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*. Birkhäuser Boston, 2004.

[13] N. Trichakis, A. Zymnis, and S. Boyd. Dynamic Network Utility Maximization with delivery contracts. *Proceedings IFAC World Congress*, pages 2907–2912, 2008.

[14] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed Newton method for Network Utility Maximization, I: Algorithm. *LIDS Report 2832, submitted for publication*, 2011.

[15] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed Newton method for Network Utility Maximization, II: Convergence. *LIDS Report 2870, submitted for publication*, 2011.